Indian Institute Of Technology Madras

ID5130 Parallel Scientific Computing

Assignment 2

AE20B027 Hareesh P Nair

Question 1

Consider the following one-dimensional first-order traveling wave equation that convects to the right,

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \tag{1}$$

with $c = 1.0$ in a domain of length $L = 2.0$. The initial condition for the wave is given as

$$u(x, t = 0) = u_0(x) = \left\{ \begin{array}{ll} sin(4\pi x), & \text{for } 0 \leq x \leq 0.5 \\ 0, & \text{for } 0.5 \leq x \leq 2 \end{array} \right\}$$

The exact solution at any time instant, t, is given by $u(x, t) = u_0(x - ct)$. Consider the numerical solution of this equation using Euler explicit time integration method and two different spatial discretization schemes (i) first-order upwind scheme and (ii) third-order accurate QUICK scheme. The Euler explicit time discretization is given as follows:

$$\frac{\partial u}{\partial t} \approx \frac{u_i^{n+1} - u_i^n}{\Delta t} \tag{2}$$

and the first-order upwind scheme is given by,

$$\frac{\partial u}{\partial x} \approx \frac{u_i^n - u_{i-1}^n}{\Delta x} \tag{3}$$

and the QUICK scheme is given by,

$$\frac{\partial u}{\partial x} \approx \frac{1}{\Delta x}\left(\frac{3}{8}u_i^n - \frac{7}{8}u_{i-1}^n + \frac{1}{8}u_{i-2}^n + \frac{3}{8}u_{i+1}^n\right) \tag{4}$$

where Δx is the grid spacing and Δt is the time-step and they can be taken as $\Delta x = 0.002$ and $\Delta t = 0.0001$. The boundary conditions on the left and right-ends of the domain are given as $u(x = 0, t) = 0$ and $u(x = L, t) = 0$. Use upwind scheme for the near boundary point ($x = \delta x$) where QUICK scheme cannot be applied.

(a) Develop a serial program to solve this problem using upwind and QUICK schemes. Plot the analytical solution and the numerical solution obtained at several times $t = 0, 0.5, 1.0$ in the same graph.

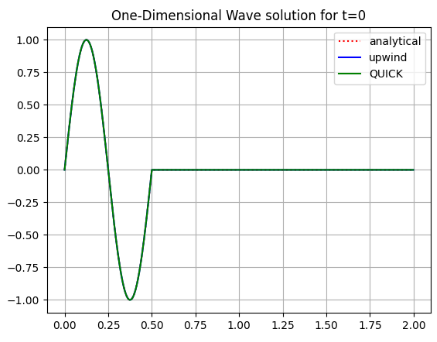
(b) Develop an MPI program to solve this problem using both the spatial discretization schemes as above. Plot the analytical solution and the numerical solutions at times $t = 0, 0.5, 1.0$ in the same graph for number of threads $p = 2and4$. Use appropriate number of halo/virtual points on both sides of the parallel domains.

(c) Comment on the differences observed between the solutions for upwind and QUICK schemes.

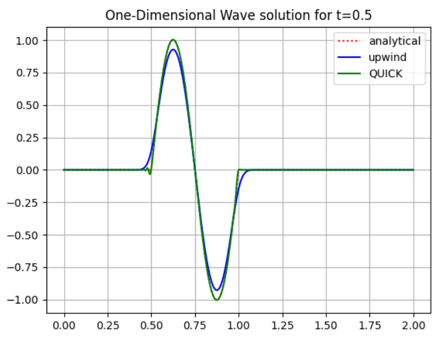
Solution

Part - a

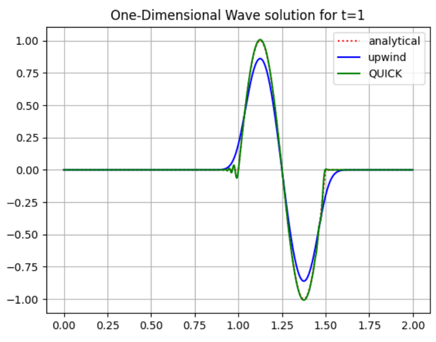
The serial code for the one dimensional travelling wave was developed and the resulting plot for the analytical and numerical solutions were obtained.



(a) time = 0



(b) time = 0.5



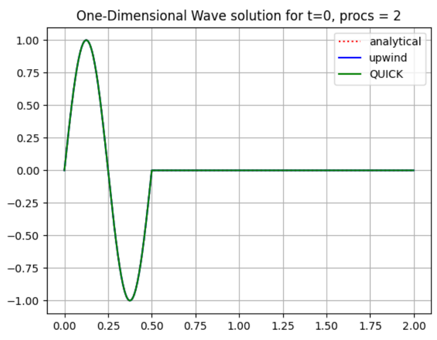
(c) time = 1

Figure 1: Plots for serial 1D travelling wave problem

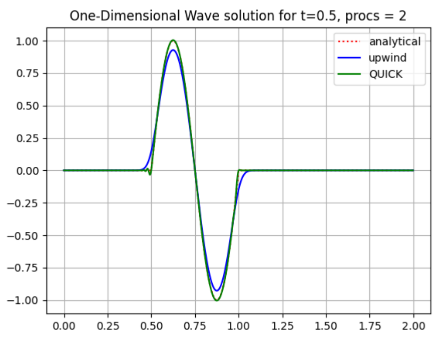
Part - b

An MPI program was developed to solve the one-dimensional traveling wave equation. The structure of the MPI program is that the domain $x = 0$ and $x = 2$ was split among the processors equally. Each of the processor runs the calculation till the time value which is incremented by 0.0001 has reached the maximum time limit 1. For each time instance, the processor of rank 0 collects the values from the other processors after the calculation the wave height and add its to the solution matrix.

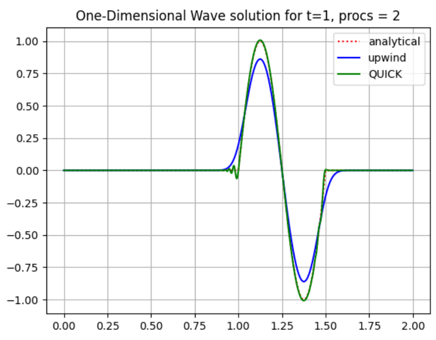
- The analytical solution part did not require any halo points as the calculation is purely dependent only on the time value.
- The upwind scheme required one halo point to be sent from the $(i-1)^{\text{th}}$ processor to i^{th} processor, where i varied from $1 \dots p$, p being the number of processors used for calculation.
- The QUICK scheme required one halo point to be sent from i^{th} processor to $(i-1)^{\text{th}}$ processor and 2 halo points to be send from $(i-1)^{\text{th}}$ processor to i^{th} processor.



(a) time = 0

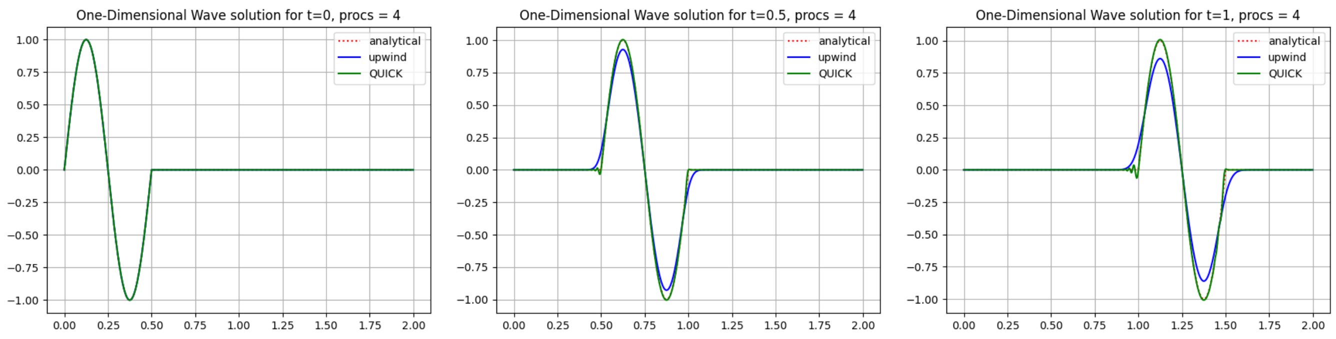


(b) time = 0.5



(c) time = 1

Figure 2: Plots for MPI (2 processors) program 1D travelling wave problem



(a) time = 0 (b) time = 0.5 (c) time = 1

Figure 3: Plots for MPI (4 processors) program 1D travelling wave problem

Part - c

As we can see from the images above, in all the three cases, serial, running on 2 processors and 4 processors, both the scheme provide results with almost accurate value, especially. However, as time instance changes, the following points are observed

- The upwind scheme diverges from the analytical solution as time in incremented.
- The QUICK scheme shows a hint of unstableness near the start and end of the wave. But the apart from that, it has proven to be very accurate.

Question 2

Consider the solution of the following Poisson equation,

$$\nabla^2\phi = -q; \quad q = (x^2 + y^2); \tag{5}$$

in the domain shown in Figure 1 and the boundary conditions are as specified in the figure.

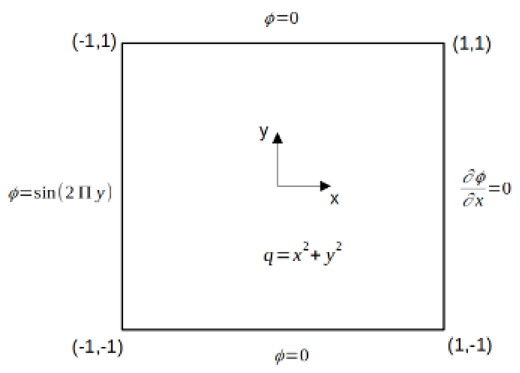


Figure 4: Domain and boundary condition for Poission equation

The discretized equation using Jacobi or Gauss-Seidel method can be written as follows,

$$\phi_{i,j}^{(k+1)} = \frac{1}{4}\left[\phi_{i+1,j}^{(k)} + \phi_{i-1,j}^{(k),(k+1)} + \phi_{i,j+1}^{(k)} + \phi_{i,j-1}^{(k),(k+1)}\right] + \frac{\Delta^2}{4}q_{i,j} \tag{6}$$

where $\Delta = \Delta x = \Delta y$. Consider an initial guess of $\phi(x, y) = 0$ everywhere. Use double precision arithmetic. On the $x = 1$ boundary, approximate the first derivative $\frac{\partial\phi}{\partial x}$ using the second-order accurate one-sided formula as given below,

$$\phi_i = \frac{4\phi_{i-1} - \phi_{i-2}}{3} \tag{7}$$

(a) Develop a serial program using Jacobi iterative method. Test your program using $\Delta = \Delta x = \Delta y = 0.1$ (that is 21 points along each of the directions). Report the number of iterations for the solution between successive iterations to converge to 10^{-4}, using any norm. Plot the numerical solution of ϕ vs x for $y = 0.0$ and ϕ vs y for $x = 0.0$.

(b) Develop an MPI program using Jacobi iterative method. Test your program using $\Delta = \Delta x = \Delta y = 0.01$ (that is 201 points along each of the directions). Report the number of iterations for the solution between successive iterations to converge to 10^{-4}, using any norm. Plot the numerical solution obtained using the Jacobi iterative method run on parallel system using $p = 2, 4, 8$ for ϕ vs x for $y = 0.0$ and ϕ vs y for $x = 0.0$. Use a one-dimensional domain decomposition (either row-wise block decomposition or column-wise block decomposition on the two-dimensional mesh) only. In the same plots compare the parallel solution obtained with the serial solution for the same grid sizes of 0.01.

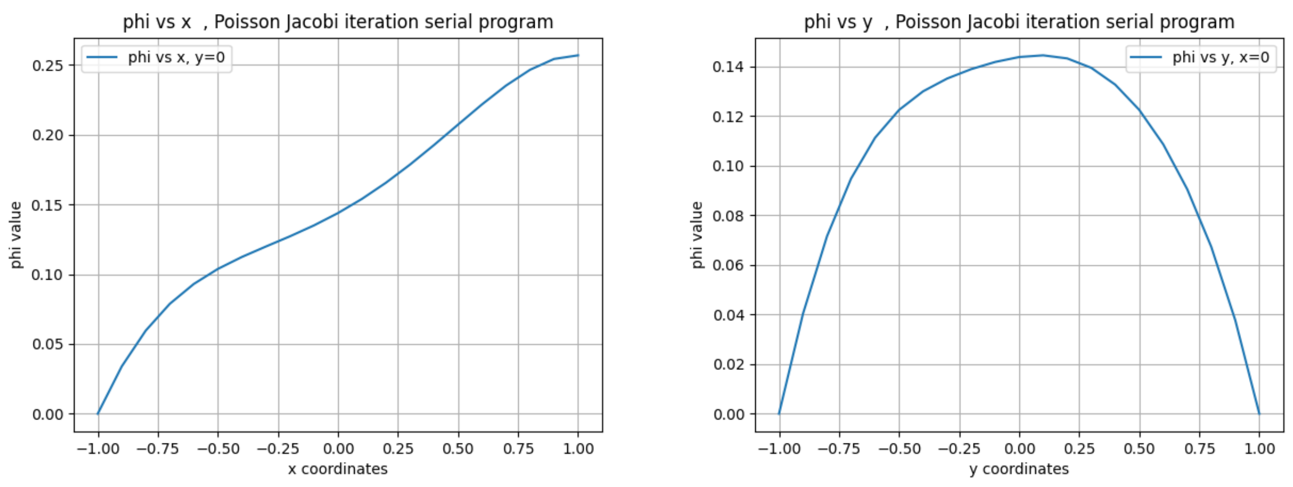
(c) Repeat the above step using Gauss-Seidel red-black coloring approach. Comment on the number of terations required by Gauss-Seidel when compared the Jacobi method.

(d) Run parallel versions of the Jacobi and Guass-Seidel methods for $\Delta = 0.005$ using $p = 2, 4, 8$ and 16 threads. Plot the speed-up, $\psi(n, p)$, obtained by each of the methods as a function of the number of processors. Do you see any improvement in performance? Which method is better? Comment on the speedup obtained as the problem size is increased.

Solution

Part - a

The serial program for the Jacobi iteration for solving the Poisson equation was developed and the following results were obtained.

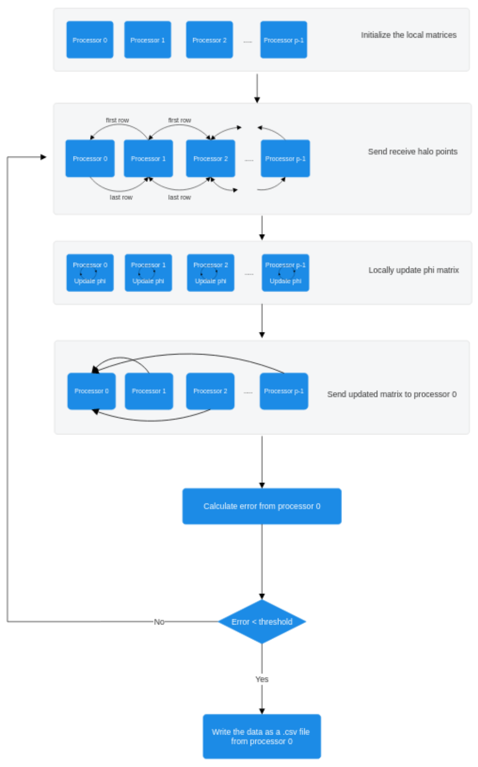


(a) ϕ vs x for $y = 0$ (b) ϕ vs y for $x = 0$

Figure 5: Plots for Jacobi iteration serial program for grid size, $\Delta = 0.1$

Part - b

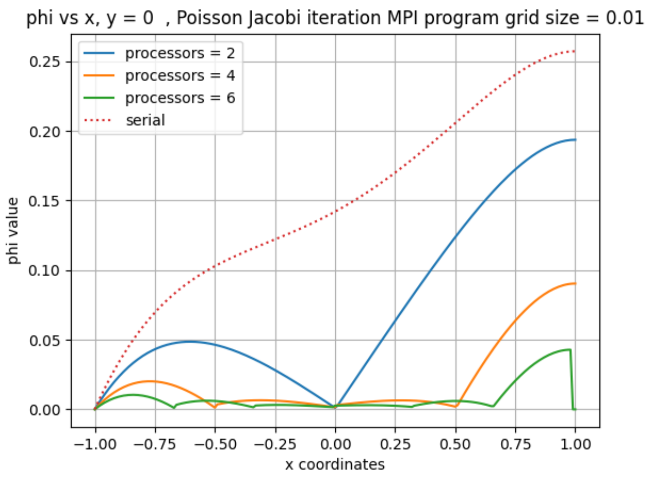
The MPI program for the Jacobi iteration for solving the Poisson equation was developed and the following results were obtained. The domain was decomposed using row decomposition. The workflow of the MPI program is as shown below:



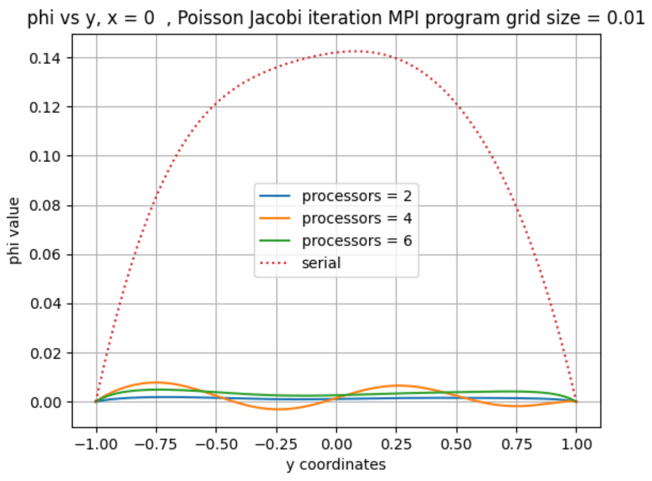
Processor 0
Processor 1
Processor 2
....
Processor p-1
Initialize the local matrices
first row
first row
last row
last row
Send receive halo points
Update phi
Locally update phi matrix
Send updated matrix to processor 0
Calculate error from processor 0
Error < threshold
No
Yes
Write the data as a .csv file from processor 0

Figure 6: MPI program workflow

The results obtained from the serial and parallel program for $\Delta = 0.01$ is as shown below:



(a) ϕ vs x for $y = 0$



(b) ϕ vs y for $x = 0$

Figure 7: Plots for Jacobi iteration MPI program for grid size, $\Delta = 0.01$

The error for the program using different number for processors was plotted against the number of iterations.

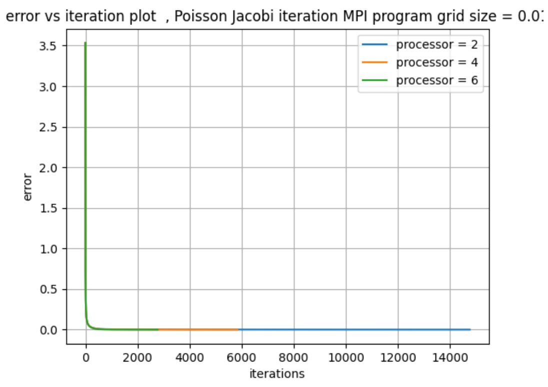


Figure 8: Error vs number of iterations for MPI Jacobi iterations ,$\Delta = 0.01$

The number of iterations taken for the program to complete using different number of processors are displayed in the table below:

S.No	**Processors**	**Iterations**
1	2	14752
2	4	5842
3	6	2775

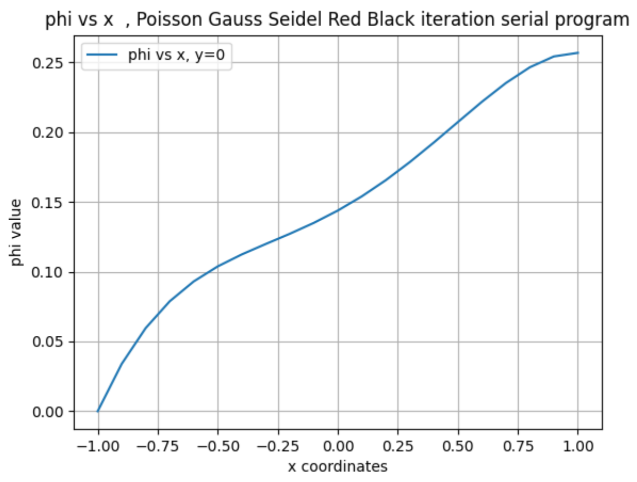
Table 1: Number of iterations taken for each number of processors for Jacobi iterations , $\Delta = 0.01$

Info : Due to my device's limits, I could only run the calculations for a maximum of 6 number of processors

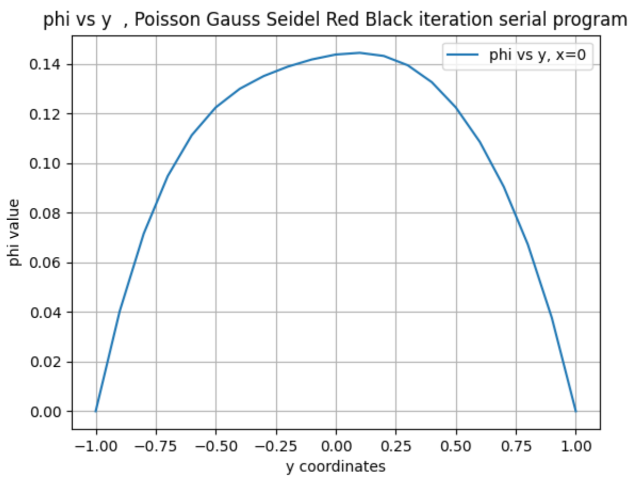
Part - c

The serial and MPI program for the Gauss Seidel red-black coloring method was developed and the following results were obtained

Serial program



(a) ϕ vs x for $y = 0$

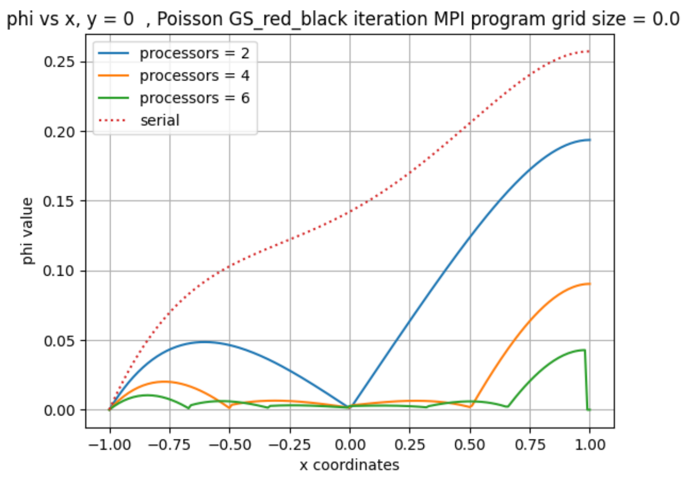


(b) ϕ vs y for $x = 0$

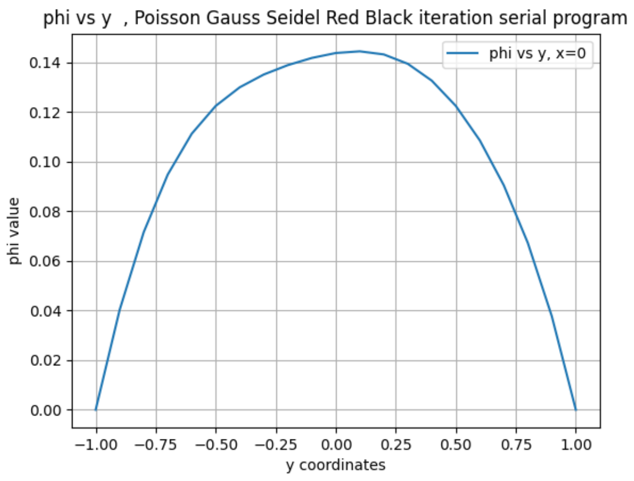
Figure 9: Plots for Gauss Seidel iteration serial program for grid size, $\Delta = 0.1$

MPI program

The workflow of the MPI program for the Gauss Seidel method is similar to that of the Jacobi iterations. The following plots were obtained



(a) ϕ vs x for $y = 0$



(b) ϕ vs y for $x = 0$

Figure 10: Plots for Gauss Seidel iteration MPI program for grid size, $\Delta = 0.1$

Similar to the Jacobi iteration, the error vs the number of iterations were plotted and the following figure was obtained.

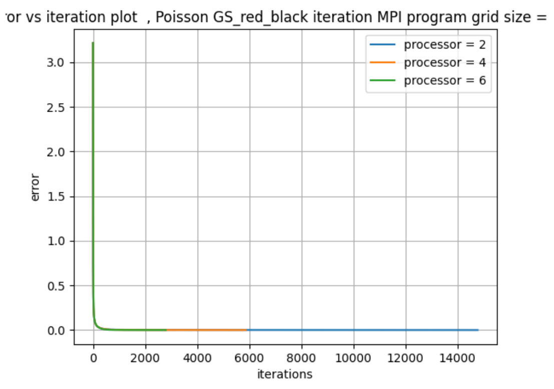


Figure 11: Error vs number of iterations for MPI Gauss Seidel iterations ,$\Delta = 0.01$

The following number of iterations were taken for the different number of processors:

S.No	**Processors**	**Iterations**
1	2	14777
2	4	5863
3	6	2790

Table 2: Number of iterations taken for each number of processors for Gauss Seidel iterations , $\Delta = 0.01$

Part - d

The time taken for the serial part of the program and the total program was calculated. Using those values the speedup $\psi(n, p)$ was calculated and plotted against different number of processors. The following plot was obtained

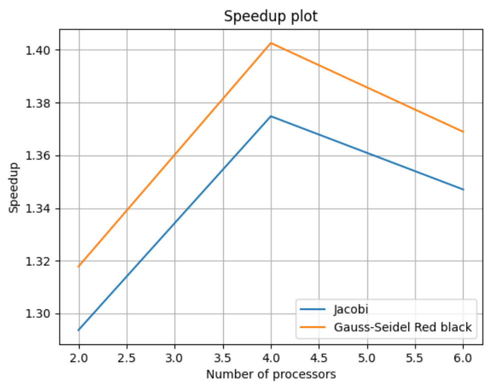


Figure 12: $\psi(n, p)$ vs p for $\Delta = 0.01$

From the speedup plot obtained, we can see that the speedup increased for number of processors, $p = 4$. However a decline can be seen upon further increase in the number of processors. The reason for that would be due to the specific workflow that is being followed in program (calculation of error from 0^{th}) processor. If that also had been split across processors, then the speedup would have increased further.
From the speedup plot, we can confidently say that Gauss Seidel red black coloring method is much better than the Jacobi iteration method.
From running the program for different values of Δ, we can see that the speedup improves for the same value of the number of processor used.